An Algorithm for the Maximum Revenue Jobshop Problem

Michal Penn
Faculty of Industrial Engineering and Management
Technion, Haifa, Israel
and
Tal Raviv
Faculty of Engineering, Department of Industrial Engineering
Tel Aviv University, Israel

July, 2007

Abstract

In this paper we state and study the problem of selecting a product mix and a dispatching rule for a jobshop system in order to maximize its revenue rate over an infinite planning horizon.

We solve the problem of obtaining maximum revenue by selecting a product mix and determine a schedule. The designed algorithm appropriately rounds an optimal solution to a fluid relaxation in which we replace discrete jobs with the flow of a continuous fluid. The algorithm solves the fluid relaxation optimally and then aims to keep the discrete schedule close to the continuous one obtained by the fluid solution. The schedule obtained is cyclic, with bounded WIP and asymptotically optimal.

A secondary aim is to further reduce the WIP and the buffers' sizes by shortening the cycle length. This is achieved at the cost of a slight compromise on the revenue. We report on satisfactory computational results on some benchmark instances.

1 Introduction

In this paper we state and study the problem of selecting a product mix and a dispatching rule for a jobshop system in order to maximize its revenue over an infinite planning horizon. More precisely, we consider the *Infinite Horizon Maximum Revenue Job Shop Problem* which is the following jobshop like problem. A set of R potential product types can be produced using a set of M machines. The revenue obtained from the system is given by a general function of the production rates of the various products. The aim is to determine a product mix and a schedule that maximizes the long run revenue rate obtained from the system. This problem was motivated by problems faced in, for example, semiconductor production lines, where a manufacturer owns a network of flexible machines and his objective is to maximize the revenue while attempting at keeping low production costs.

Our solution approach follows a growing body of research regarding the applications of fluid view for the optimization and analysis of discrete flow system. Bertsimas and Gammarnik (1999) and Bertsimas and Sethuraman (2002) have devised algorithms that approximate the minimum makespan in the jobshop problem within an additive constant from the lower bound C_{max} which is the total load on a most loaded machine. This constant depends on the total number of product types, the length of the routes and the processing times of the operations on the machines but not on C_{max} itself. Hence as the number of jobs of each type grows the ratio between the makespan of the obtained solution and the lower bound converges to one. Boudoukh et al. (2001), have presented a cyclic schedule for a high multiplicity version of the jobshop problem where the number of jobs of each type r to be processed is $n \cdot j_r$ with the j_rs being integers and n, a large integer number. Their solution is based on the idea of building safety stocks of $\lceil \frac{j_r}{2} \rceil$ units in the system buffers. They provided a schedule that, by utilizing the above safety stocks, works at the same rate as a fluid counterpart system. Thus, the approximation is within the constant time needed to create the required safety stocks and to empty the system at the end. This method is particularly effective in the typical situation where n is large while the j_rs are small. Boudoukh et al. (2001) also devised a dispatching rule, based on the fluid solution, that is shown numerically to be effective for high multiplicity jobshop problem with random processing times under the makespan objective function.

Dai and Weiss (2002) considered the problem of minimizing the makespan in jobshop setting with random processing times where there are N identical jobs in each route. They proposed a simple dispatching heuristic and proved that under some assumptions on the distribution of the processing times, the obtained solution exceeds the optimal one by at most $c \log N$ for some constant c, with probability higher than $1 - 1/N$.

Bertsimas et al. (2003) provided an algorithm for approximating the high multiplicity jobshop problem with holding cost objective based on relaxed fluid solution of the problem. They have shown that the algorithm is asymptotically optimal and performs better than many known heuristic methods on benchmark problems, even for problems with moderate degree multiplicity.

A closely related branch of literature focuses on methods to solve hard instances of the fluid control problem. Such solutions can then be used by the above mentioned methods to approximate jobshop problems and for other applications, e.g., control of queuing networks. Two recent representatives of this trend are Weiss (2001) and Fleischer and Sethuraman (2003).

The problem discussed in this paper differs from the ones studied in the above mentioned papers in the sense that the actual product mix to be produced is not exogenously given but is to be decided within the model. We show that an optimal value of the fluid version of the problem can be attained, asymptotically, using our proposed dispatching policy on the discrete system providing that sufficient safety stocks are initially accumulated. Our dispatching policy is closely related to the Revised Fluid Synchronization Algorithm proposed in Bertsimas et al. (2003) and the idea of initially creating some safety stocks follows Boudoukh et al. (2001). That is, the obtained schedule starts by building up a sufficient level of safety stocks in the system's buffers and then continue in cyclic manner while keeping the bottleneck machine busy continuously.

Indeed, we show that our dispatching rule leads to a cyclic schedule, but, unfortunately, the number of products per cycle may be arbitrarily large. We then show that the required safety stocks can be bounded above by a constant which does not depend on the cycle length, but only on the known processing times of the operations. However, our upper bounds may still lead to fairly large safety stocks.

A further reduction of the safety stocks and WIP can be gained by shortening the cycles. We achieve this goal by modifying the optimal product mix solution in a way that slightly compromises on the optimality of the product mix while significantly reduces the cycle length. A compact Mixed Integer Linear Program (MILP) is solved in order to construct this new short cycle.

We examine our method on a large set of benchmark problems and show that it is possible to construct solutions with revenue rate of 99% of the optimum with cycles consisting of very few items.

In Boudoukh et al. (2001) it is shown that the safety stock needed for each operation in a cyclic schedule is bounded above by approximately half of the items per cycle. Hence reduction of the cycle length is likely to result in a schedule that requires small amount of safety stock. Furthermore, in this study, we show that the total level of WIP is closely related to the total amount of safety stocks.

In addition, with shorter cycles it is possible to employ exact combinatorial optimization methods to further reduce the required safety stock and the average levels of intermediate inventory.

The rest of this paper is organized as follow: In Section 2 we provide formal definition of the problem and presents its fluid relaxation. In Section 3 we construct our fluid based dispatching rule, explore its cyclic nature and prove its asymptotic optimality using the relation between the fluid and the discrete solutions. In Section 4 we show how to construct an approximately optimal schedule with shorter cycle. The practicality of our method is supported by extensive numerical experiments presented in Section 5. We conclude with a short discussion in Section 6.

2 Problem Definition and Fluid Approximation

We consider the following Infinite Horizon Maximum Revenue Job Shop Problem which is a jobshop like problem over an infinite planning horizon. Consider a set of R product types and a set of M different machines. The production process of a product of type r consists of a series of K_r operations that should be carried out on a subset of the machines according to a predetermined order. We refer to this order as the *product route* and to an unfinished product as a *job*. That is, a completed job is a *product*. We allow re-entrant policy, that is, any machine may appear several times along a product route.

In our model each machine works on at most one job at a time and a product can be processed by at most one machine at a time. We consider non-preemptive policy, that is, an operation cannot be aborted before it is finished. The aim is to construct a schedule that maximizes the revenue rate achieved if the system is to be operated **indefinitely**.

The revenue rate of the firm, assuming all its production activity is carried out by the system, is a function of its production rate for each of its products. Let us denote the average production rate for product r by x_r. For a price taker firm in a perfectly competitive market, if the market price of product r is P_r then the revenue rate is clearly $\sum_{r=1}^{R} x_r P_r$. More generally the total revenue is a function $f : \mathbb{R}^R \to \mathbb{R}$. Such a function can capture market power if $\frac{\partial^2 f(\mathbf{x})}{\partial^2 x_r} < 0$ (i.e., decreasing marginal revenue), $\frac{\partial^2 f(\mathbf{x})}{\partial x_r \partial x_q} < 0$ for each pair $r \neq q$ of substitutional products and $\frac{\partial^2 f(\mathbf{x})}{\partial x_r \partial^2 x_q} > 0$ for each pair of complementary products. We can take $f(\mathbf{x})$ as the *gross profit function*, i.e., the revenue net of variable production costs, such as labor and raw material. In such a case $f(\mathbf{x})$ may capture economics of scale. Note however $f(\mathbf{x})$ does not include costs that are affected by scheduling decisions, such as holding costs of work in process.

The o^{th} operation of route r is denoted by (r,o) and the time needed to carry out operation (r,o) is denoted by $T_{r,o}$. The machine that processes operation (r,o) is denoted by $M_{r,o}$. Let σ_i be the set of operations carried out by machine i. $\sigma(r,o)$ denotes the set of all operations carried out by the same machine as operation (r,o) and $\sigma(r,o)^- \equiv \sigma(r,o) \setminus \{(r,o)\}$. Consider the jobs on route r that by time t have completed operation $(r,o-1)$ but have not yet started operation (r,o). We say that these jobs are located in the *buffer of operation* (r,o) and we denote their number by $B_{r,o}(t)$. We note that according to our definition there is no buffer before operation $(r,1)$ for each route r. Naturally, $B_{r,o}(t) \geq 0$. However, for methodological purpose we initially assume that $B_{r,o}(t)$ is allowed to go negative and later on remove this assumption.

Let $B^+_{r,o}(t)$ be the total number of jobs of type r which by time t have completed operation $(r,o-1)$ but have not yet completed operation (r,o). That is, $B^+_{r,o}(t) \in \{B_{r,o}(t), B_{r,o}(t)+1\}$. We set $B^+_{r,1}(t) = 1$ if the first operation of product of type r is processed at time t, and $B^+_{r,1}(t) = 0$ otherwise. Jobs within the system are called Work In Process (WIP). The total WIP in the system at time t is $\sum_{r=1}^{R}\sum_{o=1}^{K_r} B^+_{r,o}(t)$.

Next, we define a fluid counterpart system of the discrete system described above. In this system along each product route, fluid is continuously processed. The flow rate of fluid of type r at stage (r,o) when machine $M_{r,o}$ devotes all its effort to process it is $\frac{1}{T_{r,o}}$. However, in the fluid system the machines capacity and the "jobs" are infinitely devisable. Our fluid system start its operations with empty buffers. As in the discrete model, a revenue rate is associated with outflow rate of the system.

The goal of the auxiliary fluid jobshop problem is to maximize the revenue rate obtained from the fluid that comes out of the system. Note that an optimal solution of the fluid problem can be achieved without safety stock and WIP.

The solution to the fluid problem is given by the following simple Mathematical Program:

Mathematical Program 2.1

$$\max f(x_1, ..., x_r)$$

$$\sum_{(r,o)\in\sigma_i} T_{r,o} x_r \leq 1 \qquad \forall i = 1, ..., M \tag{1}$$
$$x_r \geq 0 \qquad \forall r = 1, ..., R.$$

Mathematical Program 2.1 can be a challenging optimization problem if the function $f(\mathbf{x})$ is not concave or not smooth. On the other hand if $f(\mathbf{x})$ is linear then the problem is reduced to the classical product mix model. Solution procedure for this problem is outside the scope of this study. In the sequel we will assume that an arbitrarily accurate *rational* approximation $\mathbf{x}^*$ for the solution of the problem is given. We note however that an optimal solution always exists since the feasible solution set of 2.1 is compact for all non negative T_{ro}.

For the convenience of the reader we added a list of notations in Appendix B.

3 The Fluid Based Dispatching Rule Procedure

In this section we present our main result, a method to construct an infinite schedule for the discrete system based on the optimal solution of its fluid counterpart. Our schedule consists of two phases: The *initialization phase* and the *main phase*. The initialization phase is finite and can be described by the starting times of each operation on its machine. The infinite main phase is described by a *dispatching rule*, that is, a method to decide upon the next operation of each machine whenever it is ready. A machine is said to be *ready* at time t if it can start to process a job at that time.

Given our proposed dispatching rule for the main phase, the purpose of the initialization phase is to build sufficient safety stocks to allow smooth operation of the system during the main phase. That is, to enable the operation of the dispatching rule such that whenever it tries to dispatch a job, this job is available in its buffer. The number of jobs accumulated in the buffers, at the end of the initialization phase, is called *safety stock*. The problem of calculating the minimum required safety stocks is discussed in the sequel. For given required safety stocks, the initialization phase is equivalent to the minimum makespan jobshop problem. This is an NP-hard and well known problem that is not studied in this paper. Here we focus on the dispatching rule for the main infinitely long phase. For methodological end, we will first assume that no initialization is carried out and that the levels of the system buffers are allowed to go negative. Later on, we will show that there exist finite levels of safety stocks that enable to remove the above assumption.

We use the argument t to denote the time passed since the beginning of the main phase. Thus, for each operation (r, o), the level of its safety stock used is $B_{r,o}(0)$. Let $D_{r,o}(t)$ denote the total number of operations of type (r, o) that were completed during the time interval $[0, t]$.

Recall that in the fluid counterpart system, each operation (r, o) is being processed at a constant rate x_r. Thus, during t units of time, $t \cdot x_r$ units of fluid are transferred along route r.

Definition 3.1 *The* ***lateness*** *of operation* (r,o) *at time* t *is* $\mathcal{L}_{r,o}(t) = t \cdot x_r - D_{r,o}(t)$.

Note that at each point of time t, $\mathcal{L}_{r,o}(t)$ represents the difference between the amount of fluid that was transferred along route r by time t in the fluid system, and the number of jobs that completed operation (r,o) in the discrete system during the main phase. We are now ready to present our proposed dispatching rule to control the operation of the discrete system during the main phase.

Definition 3.2 (Fluid Based Dispatching Rule - FBDR) *At any given time* t, *if machine* i *is ready and* $\max_{(r,o)\in\sigma_i} \mathcal{L}_{r,o}(t) \geq 0$, *then process a job from a buffer* (r,o) *where this maximum attained. Otherwise, machine* i *stays idle until the situation is changed. Ties are broken in favor of operations with the largest value of* x_r*; if few operations have equal value of* x_r, *then the ties are broken by lexicographic order of the corresponding* (r,o)*'s.*

We note that this dispatching rule is similar to the Revised Fluid Synchronization Algorithm (RFSA) proposed by Bertsimas et al. (2003) for the finite deterministic case and to the Greedy Tardiness Rule proposed by Boudoukh et al. (2001) for the finite stochastic case. The following lemma shows that under FBDR, the total amount of work, in terms of working time, done by the discrete system over all machines, is at least as the amount of work done by the fluid system.

Lemma 3.3 *Consider a discrete system that operates under FBDR assuming buffer levels are allowed to go negative. Then,* $\sum_{(r,o)\in\sigma_i} \mathcal{L}_{r,o}(t) T_{r,o} \leq 0$, *at any time* t *when machine* i *is ready.*

Proof. We first show that the lemma holds whenever the machine is idle. Assume by contradiction that the lemma does not hold for this case. Then, there must be at least one operation $(r,o) \in \sigma_i$ and a time t, when the machine is idle and $\mathcal{L}_{r,o}(t) > 0$. Now, by FBDR, the machine should have started performing operation (r,o) on an available job in the buffer (and there is always an available job in the buffer since we allow the buffer level to go negative). This contradicts the idleness of the machine.

Assume a point of time t_2 when the machine is ready but not idle. That is, an operation has just finished being processed on the machine and the machine is about to start processing its next operation. Let t_1 denote the last time the machine switched from idleness to busyness (if no such time exists, let $t_1 = 0$, the starting time of the main phase). Then,

$$\sum_{(r,o)\in\sigma_i} D_{r,o}(t_2) \cdot T_{r,o} - \sum_{(r,o)\in\sigma_i} D_{r,o}(t_1) \cdot T_{r,o} = t_2 - t_1, \tag{2}$$

since the machine was busy throughout the time interval $[t_1, t_2]$. On the other hand, by constraint (1) and the fact that $t_2 \geq t_1$ we have,

$$t_2 \cdot \sum_{(r,o)\in\sigma_i} x_r \cdot T_{r,o} - t_1 \cdot \sum_{(r,o)\in\sigma_i} x_r \cdot T_{r,o} \leq t_2 - t_1. \tag{3}$$

Subtracting (2) from (3) we obtain,

$$\sum_{(r,o)\in\sigma_i} \mathcal{L}_{r,o}(t_2)T_{r,o} - \sum_{(r,o)\in\sigma_i} \mathcal{L}_{r,o}(t_1)T_{r,o} \leq 0.$$

By the first part of this proof, $\sum_{(r,o)\in\sigma_i} \mathcal{L}_{r,o}(t_1)T_{r,o} \leq 0$, implying

$$\sum_{(r,o)\in\sigma_i} \mathcal{L}_{r,o}(t_2)T_{r,o} \leq 0.$$

■

We turn now to show that using FBDR creates *cyclic schedules.* We start with a formal definition of this intuitive notion.

Definition 3.4 ***Cyclic Schedule:*** *A schedule is called cyclic with a cycle length d if at any time t, the system status is the same as in $t + d$. By system status we refer to the type of operation carried out on each machine, the elapsed times of these operations since their beginning, and the WIP level in the systems' buffers.*

We note that in a cyclic schedule, the number of products of each type delivered from the system during the interval $(t_0 + nd, t_0 + (n+1)d]$ is the same for all $n = 0, 1, 2, \ldots$.

Proposition 3.5 *The schedule obtained by applying FBDR, in its main phase, is a cyclic one.*

Proof. In the first part of this proof we will show that $\mathcal{L}_{r,o}(nd) = \mathcal{L}_{r,o}((n+1)d) = 0$ for all non-negative integers n. Then we will show that this implies the cyclic nature of the schedule obtained by FBDR.

Recall that the main phase starts at time $t = 0$ and let d be a common denominator of $x_1, ..., x_R$, where $\mathbf{x} = (x_1, ..., x_R)$ is a solution of Mathematical Program 2.1. Note that none of the $(ndx_r + 1)^{st}$ operations of route r could have been started before time nd. This is since, if for some operation (r, o), $D_{r,o}(t) = ndx_r$ at $t < nd$ (i.e., the $(ndx_r)^{th}$ operation was completed before time nd) then its lateness will remain negative until time nd. However, in such a case, the next operation would not be dispatched by FBDR. Hence,

$$\mathcal{L}_{r,o}(nd) \geq 0 \tag{4}$$

for all (r,o) and integer n. Now, if for some machine i, $\mathcal{L}_{r,o}(nd) = 0$ for all $(r,o) \in \sigma_i$, then we are done with the first part of the proof. If, however, $\mathcal{L}_{r,o}(nd) > 0$ for some (r,o) then at time nd machine i is working on one of the first ndx_r instances of (r,o). Let us denote the completion time of the (r,o) operation that is in process at time nd by t'. At this time, t', we will have

$$D_{l,u}(t') \leq ndx_l < t'x_l \tag{5}$$

for all operations $\{l,u\} \in \sigma_i$. To see why the first inequality holds consider first operations $(l,u) \in \sigma_i \backslash (r,o)$. These operations are not processed in the time interval $[nd, t']$ and hence $D_{l,u}(t') = D_{l,u}(nd) \leq ndx_l$. Now, operation (r,o), that is completed at time t', was in late at time nd. Hence $D_{r,o}(t') - 1 = D_{r,o}(nd) < ndx_r$ and by the integrality of $D_{r,o}(t)$ and of ndx_r the weak inequality follows. The second inequality in (5) holds simply by the facts that $nd < t'$ and $x_l > 0$.

Next, by (5) we have $t'x_l - D_{l,u} = \mathcal{L}_{l,u}(t') > 0$ for all $(l,u) \in \sigma_i$ and thus

$$\sum_{(l,u)\in\sigma_i} \mathcal{L}_{l,u}(t')T_{l,u} > 0.$$

But, this contradicts Lemma 3.3. Hence, it must be the case that $\mathcal{L}_{r,o}(nd) = 0$. Now by Lemma 3.3 and (4) we have that $\mathcal{L}_{r,o}(nd) = 0$ and $D_{r,o}(nd) = ndx_r$. That is, the number of the (r,o) operations that are carried out during a cycle is dx_r.

Since this is true for all the operations along route r, the number of jobs entered buffer (r,o) from machine $\sigma(r,o-1)$ at time nd, is the same as the number of jobs coming out of the buffer by machine $\sigma(r,o)$, at that time. Hence, $B_{r,o}(nd) = B_{r,o}((n+1)d)$ for all positive integers n.

That is, at time $(n+1)d$, the system state, in terms of buffer levels, machine status and lateness, is the same as at time nd and a new **identical** cycle starts. ■

The following observations follow directly from the above proof.

Corollary 3.6 *Given a system that operates under FBDR, with respect to some solution $x_1, \ldots, x_R$ of the fluid problem, we obtain a cyclic schedule such that:*

1. *The smallest common denominator of the flow rates $x_1, ..., x_R$ is the cycle length d.*

2. *The number of instances of operation (r,o) carried out during any cycle is $d \cdot x_r$.*

3. *At any point of time nd, for any $n = 0, 1, \ldots,$ all the machines are ready and $\mathcal{L}_{ro}(nd) = 0$ for all the operations (r,o).*

One simple but important implication of Corollary 3.6 is that the (negative) buffer levels are bounded from below by the number of operations per cycle. That is FBDR can be implemented without allowing negative buffer levels by applying an initialization phase in which sufficient safety stocks is accumulated in the system buffers before starting to operate the system under FBDR. In the sequel we present bounds on the required safety stocks that do not depend on the number of operations per cycle. We first show that, when applying FBDR, the *lateness* of each operation (r, o) can be bounded from above by a constant that is independent of x_r and d.

Proposition 3.7 *When operating a jobshop using FBDR and allowing negative buffer level,*

$$\mathcal{L}_{r,o}(t) < \frac{\sum_{(l,u)\in\sigma(r,o)^-} T_{l,u}}{T_{r,o}}.$$

Note that the above bound on the lateness is independent of the time t. Intuitively, Proposition 3.7 states that once a job is late on a machine it will not wait more than the time needed to process one instance of each of the other jobs that share the same machine, where the time units are measured in units that are equal to its own processing time. The formal proof is long and technical, hence, it is presented in Appendix A, where we also provide some tighter upper bounds for the lateness. We have chosen to present the above bound because of its simplicity. In the sequel we will use $\gamma_{r,o}$ to denote such an upper bound on the lateness regardless of how it was established. Next, we will show that the existence of an upper bound on the lateness implies lower bound on the buffer levels if negative levels are allowed.

Lemma 3.8 *Assume a jobshop is operated under FBDR and negative buffer levels are allowed. Then,* $B_{r,o}(t) \geq -\lceil \gamma_{r,o-1} \rceil$ *for all* t.

Proof. If the system starts with empty buffers then at time t the number of jobs that are waiting for operation (r, o) at the buffer or on the machine, is

$$B^+_{r,o}(t) = D_{(r,o-1)}(t) - D_{r,o}(t).$$

By the definition of the lateness,

$$B^+_{r,o}(t) = D_{(r,o-1)}(t) - tx_r + tx_r - D_{r,o}(t) = -\mathcal{L}_{r,o-1}(t) + \mathcal{L}_{r,o}(t).$$

Note that $B_{r,o}(t)$ decreases only at those points of time t when there is an operation (r, o) to be started on the machine. Hence, it is enough to prove the validity of our lower bound only for these points of time. Now, at these times, by the definition of FBDR, we have $\mathcal{L}_{r,o}(t) \geq 0$. Hence we have,

$$B^+_{r,o}(t) \geq -\mathcal{L}_{r,o-1}(t) > -\gamma_{r,o-1}.$$

Now, since $B^+_{r,o}(t)$ denotes the number of jobs it must be an integer, either positive or negative. Hence,

$$B^+_{r,o}(t) \geq \lfloor -\gamma_{r,o-1} + 1 \rfloor.$$

Therefore, the number of jobs waiting at the buffer is bounded below by

$$B_{r,o}(t) \geq \lfloor -\gamma_{r,o-1} \rfloor = -\lceil \gamma_{r,o-1} \rceil.$$

■

The result of Lemma 3.8 can be intuitively explained as follow: since an operation may reach its maximum lateness while the next operation down its route keeps up with the fluid solution, the maximum possible "deficit" of jobs equals the lateness of the previous machine. Using this observation, we can remove the assumption that the system buffer levels are allowed to go negative. Instead, we will start with sufficiently large safety stocks implying that the actual buffer levels remain non-negative.

Proposition 3.9 *In a system operated under FBDR, if $B_{r,o}(0) = \lceil \gamma_{r,o-1} \rceil$ for each (r, o) then $B_{r,o}(t) \geq 0$ for all t.*

Proof. Follows directly from Lemma 3.8.■

We note that the upper bound for the required safety stocks for a system operated under FBDR provided by Proposition 3.9, differs from other upper bounds obtained for previously presented dispatching rule, e.g., by Boudoukh et al. (2001), in the sense that our bound is independent of the number of operations per cycle. Below we prove that FBDR is an asymptotically optimal policy for the Infinite Horizon Maximum Revenue Job Shop Problem.

Theorem 3.10 *Let $\mathbf{x}$ be a solution of Mathematical Program 2.1. Assume a system operates under FBDR, relative to $\mathbf{x}$, with sufficient safety stocks for t units of time. Then, the revenue rate of the discrete system converges to the revenue rate of its fluid counterpart system as $t \to \infty$.*

Proof. It is enough to show that at any point of time t, the number of products of type r delivered from the discrete system by that time is within a constant (independent of t) from the amount of fluid that comes out of the corresponding route by that time. The difference between these two values is $\mathcal{L}_{r,K_r}(t) = tx_r - D_{r,K_r}(t)$, which by Proposition 3.7, is bounded above by a constant and hence implies the correctness of the theorem. Note also that even if the initialization phase is included, the theorem holds since this phase is finite. ■

Theorem 3.10 implies that by applying FBDR we maximize the revenue rate of the discrete system in the asymptotic sense. In the rest of this section we will explore the dynamics of the WIP presented in the system throughout the main phase. For this end we will construct a lower bound on the lateness.

Lemma 3.11 *Assume a system operates according to FBDR with sufficient safety stocks. Then, at any given time t and for any operation (r,o), the following inequality holds*

$$\mathcal{L}_{r,o}(t) \geq -1 + x_r T_{r,o}.$$

Proof. Recall that $\mathcal{L}_{r,o}(t)$ is an increasing function of t, except at those points of time when operation (r,o) is completed and the function decreases by one. That is, $\mathcal{L}_{r,o}(t)$ achieves its global minimum either upon the termination of an operation of type (r,o), or at the beginning where $\mathcal{L}_{r,o}(0) = 0$. Consider an instance of operation (r,o) that starts at time t_0 and terminates at time $t_0 + T_{r,o}$. Then, $D_{r,o}(t_0 + T_{r,o}) = D_{r,o}(t_0) + 1$, implying

$$\mathcal{L}_{r,o}(t_0 + T_{r,o}) = (t_0 + T_{r,o})x_r - (D_{r,o}(t_0) + 1) = \mathcal{L}_{r,o}(t_0) - 1 + x_r T_{r,o} \geq -1 + x_r T_{r,o}.$$

The last inequality is due to the fact that under FBDR, jobs are dispatched at time t_0 only if $\mathcal{L}_{r,o}(t_0) \geq 0$. ■

In other words, since under FBDR, a job is processed only when it is late as compare to the fluid system, the discrete system cannot precede its fluid counterpart by more than one job. Also observe that since for any produced product r, $x_r > 0$, we have that $\mathcal{L}_{r,o}(t) > -1$. The following proposition provides an upper bound on the level of WIP, $B^+_{r,o}$, in each particular operation (r,o) resulted by applying FBDR.

Proposition 3.12 *For all t, using FBDR with sufficient safety stocks implies that,*

$$B^+_{r,o}(t) \leq B_{r,o}(0) + 1 + \lfloor \gamma_{r,o} - x_r T_{r,o-1} \rfloor.$$

Proof. Recall that $B_{r,o}(0)$ is the amount of safety stock used and observe that at any point of time t, $B^+_{r,o}(t)$ equals the amount of safety stock plus the number of jobs completed operation $(r,o-1)$ minus the number of jobs completed operation (r,o). That is,

$$B^+_{r,o}(t) = B_{r,o}(0) + D_{r,o-1}(t) - D_{r,o}(t).$$

By the definition of the lateness, $D_{r,o}(t) = t \cdot x_r - \mathcal{L}_{r,o}(t)$ and so,

$$B^+_{r,o}(t) = B_{r,o}(0) + [t \cdot x_r - \mathcal{L}_{r,o-1}(t)] - [t \cdot x_r - \mathcal{L}_{r,o}(t)],$$

implying

$$B^+_{r,o}(t) = B_{r,o}(0) - \mathcal{L}_{r,o-1}(t) + \mathcal{L}_{r,o}(t).$$

By Lemma 3.11, $\mathcal{L}_{r,o-1}(t) \geq -1 + x_r T_{r,o-1}$, and by Proposition 3.7, $\mathcal{L}_{r,o}(t) < \gamma_{r,o}$. Hence,

$$B^+_{r,o}(t) < B_{r,o}(0) + \gamma_{r,o} + 1 - x_r T_{r,o-1}.$$

Since both $B^+_{r,o}(t)$ and $B_{r,o}(0)$ are integers, it follows that

$$B^+_{r,o}(t) \leq B_{r,o}(0) + 1 + \lfloor \gamma_{r,o} - x_r T_{r,o-1} \rfloor,$$

which completes the proof. ■

We note that the required safety stock for operation (r, o) is primarily affected by the maximum possible lateness of its preceding operation, that is, $\max_t \mathcal{L}_{r,o-1}(t)$; while the required buffer space, which is the maximum level of WIP, is affected by the lateness of both the previous and the current operations. The following proposition shows that in systems with long routes, the *total* WIP levels are approximately the same as the *total* required safety stocks.

Proposition 3.13 *When a system is operated under FBDR, the maximum fluctuation of the WIP levels from the initial safety stocks, over all buffers and machines of a given route r is bounded by*

$$-\lceil \gamma_{r,1} \rceil \leq \sum_{o=1}^{K_r} B^+_{r,o}(t) - \sum_{o=2}^{K_r} B_{r,o}(0) \leq \lceil \gamma_{r,K_r} \rceil$$

at any time t.

Proof. First, observe that at any given time, the number of jobs along any route equals: the number of jobs in the system at the beginning of the main phase (safety stocks) plus the number of jobs that started their first operation since the beginning of the phase, minus the number of jobs that completed their last operation. Observe that the number of jobs that started their first operation by time t, is either $D_{r,1}(t)$, or $D_{r,1}(t) + 1$. Therefore,

$$D_{r,1}(t) - D_{r,K_r}(t) \leq \sum_{o=1}^{K_r} B^+_{r,o}(t) - \sum_{o=2}^{K_r} B_{r,o}(0) \leq D_{r,1}(t) - D_{r,K_r}(t) + 1.$$

Hence,

$$-\mathcal{L}_{r,1}(t) + \mathcal{L}_{r,K_r}(t) \le \sum_{o=1}^{K_r} B^+_{r,o}(t) - \sum_{o=2}^{K_r} B_{r,o}(0) \le -\mathcal{L}_{r,1}(t) + \mathcal{L}_{r,K_r}(t) + 1.$$

Note that the total WIP level along a route increases only when a job starts being processed by the first machine, and that at such times $\mathcal{L}_{r,1}(t) \ge 0$. The last observation, coupled with the fact that $\mathcal{L}_{r,K_r}(t) < \gamma_{r,K_r}$ and the integrality of the total WIP, imply that

$$\sum_{o=1}^{K_r} B^+_{r,o}(t) - \sum_{o=2}^{K_r} B_{r,o}(0) \le \lceil \gamma_{r,K_r} \rceil.$$

On the other hand, for all t we have $\mathcal{L}_{r,K_r}(t) > -1$ and $\mathcal{L}_{r,1}(t) < \gamma_{r,1}$. Coupled with the integrality of the WIP level we conclude that,

$$\sum_{o=1}^{K_r} B^+_{r,o}(t) - \sum_{o=2}^{K_r} B_{r,o}(0) \ge -\lceil \gamma_{r,1} \rceil.$$

■

We note that although in many cases, the safety stocks and the WIP levels are of comparable magnitude, there are situations where this is not the case, as demonstrated by the following example. Consider a system of two machines and two product types. Product I is processed solitarily by M_2 for 100 units of time, while product II is first processed by M_1 for 2 units of time and then by M_2 for a single unit of time. Consider an optimal fluid solution, $x_1 = \frac{1}{200}$ and $x_2 = \frac{1}{2}$. If the discrete system operates under FBDR, then M_1 starts processing product II every 2 units of time, and M_2 starts with a single unit of product II, continues by processing a single unit of product I for the next 100 time units, and then processes 100 units of product II, one unit of product I, and so on. One can easily check that the required safety stocks for this policy is a single unit of product II in the buffer of M_2, while the WIP level in front of M_2 at $t = 101$ is $B_{2,2}(101) = 50$.

Corollary 3.14 *The maximum WIP* per route*, at any given time, is bounded above by* $\sum_{o=1}^{K_r} \lceil \gamma_{r,o} \rceil$.

Proof. By Proposition 3.9, the maximum required safety stock for operation (r, o) under FBDR, is $\lceil \gamma_{r,o-1} \rceil$. That is, the WIP at time zero is $\sum_{o=1}^{K_r-1} \lceil \gamma_{r,o} \rceil$. Now, by Proposition 3.13 the WIP may grow by up to $\lceil \gamma_{r,K_r} \rceil$ during the operations of the system under FBDR and so the total WIP never exceeds $\sum_{o=1}^{K_r} \lceil \gamma_{r,o} \rceil$. ■

It is worth to stress here that the actual required safety stocks and buffers' sizes for any operation under FBDR, may be lower than the bounds presented here. Thus, a better way to calculate, or estimate, the safety stock requirements is by simulation.

Due to the cyclic nature of the schedules created by FBDR, we can simulate our system for a single cycle in order to calculate the exact minimal safety stock levels and buffers' sizes required. As proposed by Boudoukh et al. (2001), we start the simulation with zero level safety stocks but allowing the buffer level to be negative throughout the cycle. The safety stock required for each buffer is the absolute value of the smallest level reached in this buffer throughout the cycle. If such safety stock levels are used, then the maximum WIP level throughout the cycle can be obtained by subtracting the minimum (negative) WIP level from the maximum WIP level in the simulation. This maximum level of WIP determines the space needed to be allocated for the system buffers. Note that the term "simulation" is somewhat misleading in this context since we are simulating the operation of a deterministic system - so this is **not** a Monte-Carlo simulation.

Note also that our schedule remains optimal under any modification of the sequence and the timing of the operations within the cycle, as long as the number of operations of each type and the cycle length are not modified. The problem of determining a cyclic schedule that minimizes the average WIP level is NP-hard already for two machines job-shop problem, see Kamoun and Sriskandarajah (1993). However, the cyclic schedule can be "improved" by various methods. See for example Hanen (1994), Lee and Posner (1997), Boudoukh et al. (2001) and Hall et al. (2002). The composition of a cycle $(dx_1, \ldots, dx_R)$ is generally referred to as Minimal Part Set (MPS) in the cyclic scheduling literature. Typically in the literature, an MPS assumed to consist of a small number of products of each type; this is unfortunately not the case for the cycles produced by our method.

4 Reducing the Cycle Length

In this section, we show how to construct a cyclic schedule for approximately optimal product mix solution with shorter and smaller MPS. We then demonstrate our method with small numerical example that illustrates the usage of the methods we proposed. Our extensive numerical experiment in the following section shows that the obtained MPS is indeed small enough for the purpose of further analysis and optimization.

A practical obstacle of taking advantage of the cyclic nature of the FBDR schedule is that in an optimal solution of Mathematical Program 2.1, the common denominator of $(x_1, \ldots, x_R)$ can be arbitrarily large and so the resulted MPS is likely to be very long. However, the cyclic scheduling literature is based on the assumption that the MPS is relatively small. Moreover, it might be even impractical to simulate one cycle of the schedule in order to check what are the actual required safety stocks. Indeed, one can resort to the upper bounds provided by Proposition 3.9. Unfortunately, these upper bounds are generally loose. Thus, a better way to overcome this difficulty may be to search for much smaller MPS by slightly compromising on the optimality of the product mix solution.

For the case of linear revenue function, $f(\mathbf{x}) = \sum_{r=1}^{R} P_r x_r$, we use the following Mixed Integer Linear Program (MILP) to obtain an MPS that yields revenue rate within a pre-specified ratio from the optimum:

Mixed Integer Program 4.1

$$\min \sum_{r\in R} K_r j_r$$

$$\sum_{(r,o)\in\sigma_i} T_{r,o} j_r \le d \qquad \forall i \in M \tag{6}$$

$$\sum_{r\in R} P_r j_r \ge \left(\delta \sum_{r\in R} P_r x_r^*\right) \cdot d \tag{7}$$

$$d \ge \epsilon \tag{8}$$

$$\mathbf{j} \in \mathbb{N}^R, d \in \mathbb{R}.$$

Here we use $\mathbf{x}^*$, an optimal solution of Mathematical Program 2.1, as data. The decision variables are j_r, which indicate how many instances of each operation along route r should be carried out during a cycle, and d the cycle length. Constraint (6) assures that the selected MPS is consistent with the cycle length. Constraint (7) assures that the ratio between the average revenue per time unit in the constructed cycle and the revenue per time unit in the fluid counterpart system, is at least δ, which is the pre-specified desired approximation ratio. Constraint (8) is aimed to eliminate the trivial solution $d = 0$ and $j_r = 0$ for all r. Without this constraint, the optimal solution of the Mixed Integer Program is always $d = 0$ and $j_r = 0$, for all r, which imply a solution with an undefined ratio of 0/0. ϵ can be chosen to be any small enough positive number. However, to enhance numerical stability, it is better to choose ϵ to be as large as possible. Thus we set $\epsilon = \min_r \max_i \sum_{(r,o)\in\sigma_i, x_r>0} T_{r,o}$. Clearly, the length of any non-empty cycle is larger than the time it takes to process each of the operations of any given route on any machine. Note that if $x_r^* = 0$ for all r, then it is indeed unworthy to produce at all. In this case, the right hand side of the second constraint of the mixed integer program vanished, and the solution obtained is $j_r = 0$ for all r and $d = \epsilon$.

Alternatively, the objective function could have been chosen to minimize the cycle length d. However, since we aim at obtaining cycles that are as easy as possible to describe and analyze numerically, it seems more beneficial to have the "length" of a cycle in terms of the number of operations and not in terms of units of time.

Using the solution of Mixed Integer Program 4.1, a new $\delta-$approximate solution of the fluid problem with $x'_r = j_r/d$ is constructed, and is used as the basis for FBDR.

Note that 4.1 is a very lean mixed integer program with only $|R|$ integer decision variables and $|M|+1$ constraints. Our numerical experiments show that in the competitive (linear) case this program can be solved for problems with few tens of routes using commercial solver in a very short time.

For the case of general (nonlinear) revenue function we can achieve the very same goal, of reducing the size of the MPS, by replacing constraint (7) with,

$$f\left(\frac{j_1}{d}, ..., \frac{j_r}{d}\right) \geq \delta f(\mathbf{x}^*).$$

A solution method for this non-linear program is out of the scope of this study but we note that if the revenue function is increasing and concave in each of its arguments then MIP 4.1 remains a convex integer program. The numerical example and experiments hereafter are all based on the linear revenue functions.

Finally we note that Mixed Integer Program 4.1 admits a feasible solution for any value of $\delta \leq 1$. In particular, letting d be a common denominator of $x_1, ..., x_R$ and $j_r = x_r \cdot d$, as in the proof of Proposition 3.5, is always a feasible solution. Nevertheless, using values of δ very close to unity, say $1 - 10^{-5}$, may result in numerical instability caused by the limited accuracy of the floating point representation used by the solver.

Numerical Example

Here we demonstrate some of the concepts presented in this paper on a small 4-4-4 jobshop problem, that is, a problem with four product types, each consisting of 4 operations on 4 different machines. For this example we assume that the firm operates in a perfectly competitive market. That is, the revenue function is linear. The relevant data is presented in Table 1 below.

product number	product path	processing times				product price
		M_1	M_2	M_3	M_4	
1	$2 \to 3 \to 1 \to 4$	4	7	12	15	27
2	$1 \to 4 \to 3 \to 2$	11	14	17	19	70
3	$4 \to 3 \to 2 \to 1$	7	15	12	8	44
4	$3 \to 2 \to 4 \to 1$	13	10	12	15	51

Table 1: Sample data of small Infinite Horizon Maximum Revenue 4-4-4 Jobshop Problem.

Solving Mathematical Program 2.1 for this problem returns;

$$x = (0.00000000001890, 0.00671140988385, 0.03355704684015, 0.04026845574536)$$

with the objective function value of about 4.02685. This result can be taken as is in order to operate the system under FBDR. Observe that only very few jobs of type 1 are produced while about 8% of the products are of type 2, 42% of type 3 and 50% are of type 4. In order to operate the system with this solution, one can obtain upper bounds on the required safety stocks by employing the results of Proposition 3.9 and Proposition A.1 (in the appendix). These bounds are presented in Table 2 below.

The smallest common denominator of the values x_r presented above is clearly huge (greater than 10^{11}). Generally, the common denominator cannot be calculated accurately and its order of magnitude is mainly determined by the accuracy of the Linear Programming solver. As discussed above, it might be better to use a slightly suboptimal solution of Mathematical Program 2.1 with x_r's that admit much smaller common denominator and yield schedules with shorter cycle lengths. We use Mixed Integer Program 4.1 to find the shortest possible cycle that guarantees approximation ratio of say $\delta = 0.99$. For our demonstration problem, the solution for this program is $j_1 = 0, j_2 = 1, j_3 = 2, j_4 = 2$. That is, each cycle consists of one product of type 2, and two products of types 3 and 4 each. No production of product of type 1 takes place. The cycle length for this case is $d = 65$ compares to $d > 10^{11}$ above. The expected revenue per time unit is about 4.0154, which is approximately 0.997 of the optimal solution. One possible schedule for this cycle is the one induced by FBDR with $\mathbf{x} = (\frac{j_1}{d}, \frac{j_2}{d}, \frac{j_3}{d}, \frac{j_4}{d}) = (0, \frac{1}{65}, \frac{2}{65}, \frac{2}{65})$.

The safety stocks required for this schedule are easily calculated by simulating the system. Table 2 presents the actual required safety stocks for the problem presented in Table 1 against the theoretical upper bounds obtained by Proposition 3.9.

product	Bounds			Actual			
	Safety Stock			Safety Stock			Products
number	o=2	o=3	o=4	o=2	o=3	o=4	per Cycle
r=1	1	1	1	-	-	-	0
r=2	1	1	1	0	1	1	1
r=3	3	2	2	0	1	1	2
r=4	4	4	4	1	1	1	2

Table 2: Comparison between the upper bounds on the required safety stocks for the original fluid based solution and the actual safety stocks needed for the short cycle obtained by MIP 4.1 with $\delta = 0.99$.

5 Numerical Study

In this section we demonstrate the practicality of the methods described in the paper using some famous benchmark jobshop problems, extended with a list of prices of the products. We constructed two 10x10 and four 20x20 test problems, based on Adams et al. (1988) (denoted by 'abz5' and 'abz6') and Yamada and Nakano (1992) (denoted by 'yn1',...,'yn4'), respectively. The test data sets were downloaded from OR-Lib at "http://mscmga.ms.ic.ac.uk/jeb/orlib/jobshopinfo.html". In addition, since all OR-Lib problems are of non re-entrant lines, while our approach treats re-entrant lines as well, we generated three new problems. Each of the new problems consists of 10 machines, 20 product types, and each product route consists of 40 operations where each product may visit the same machine several times, but any pair of consecutive operations are carried out by different machines. The processing times of all operations in these examples were sampled uniformly from the sets $\{10, \ldots, 49\}$, $\{1, \ldots, 99\}$ and $\{1, \ldots, 999\}$. These problems are denoted by 'pr1','pr2' and 'pr3', respectively.

We believe that FBDR works well in cases where the processing times of the various operations are of similar magnitude. Thus we "cooked" an extreme example, denoted by 'ext' in order to examine the performance of our method in such extreme cases. Half of the jobs in 'ext' are "light" ones, while the other half are "heavy". The processing times of all light jobs, for all their operations, are uniformly sampled from the set $\{1, \ldots, 10\}$, while the processing times of all operations of the heavy jobs are uniformly sampled from the set $\{990, \ldots, 999\}$. As we expected, in such problems, the accumulated WIP is relatively large and hence they may not fit the fluid approach very well. This is because during the time a heavy job is processed, many light jobs may be accumulated in front of it, and thus create relatively large WIP. Note however that this excessive WIP consists mainly of the lighter jobs that are also likely to be cheaper to maintain within the system.

In this experiment we assume that the revenue function is linear. Hence, for each of these ten jobshop problems we randomly generated 20 price vectors. Each such vector assigns a price for each product type. The prices were taken from the Normal distribution with mean $\mu_r = \sum_{o=1}^{K_r} T_{r,o}$ and σ^2 equals the variance of the processing times over all operations of route r, rounded to the nearest integer. By using prices proportional to the total processing times of the jobs, we created instances with relatively many product types to be produced in the optimal product mix, and avoid trivialization of the problem.

For each of the 200 test problems, we calculated the optimal fluid solution using Mathematical Program 2.1 and then, use MIP 4.1 to obtain the shortest possible solution that delivers revenue of at least 99% of the optimum. The MIPs were solved using CPLEX 8.0 on a mobile Pentium III , 1GHz with 384Mb RAM. The solution times of the programs solved ranged from fraction of a second to very few seconds in the worst case. Our test data is available from our site at "http://eng.tau.ac.il/∼talraviv/Publications". Once we obtained the approximated solution, we translated it back into a suboptimal solution of the fluid problem, and use this solution to create the appropriate FBDR policy for each of the problems. We then simulate a single cycle for each such problem, to find out the required safety stocks, buffer spaces and the average level of WIP. These results are summarized in Table 3. Each line presents the average over the 20 different problem instances, with different products prices.

In column 1 of Table 3 we present the name of each problem. Column 2 indicates the dimensions of the problem as triplet *(#machines - #product types - #operation per products).* In column 3, the range of the processing times for all operations is presented. In column 4 we present the average number of product types that were actually produced while using the optimal solution of MIP 4.1, with $\delta = 0.99$. In both solution approaches, and for virtually all the problems, the number of product types that were actually produced ($j_r > 0$) was significantly less than R, the number of available routes. In column 5 we present the average number of products per cycle. Note that we present the cycle length in terms of the number of products, and not in terms of the number of operations, in order to make a fair compression between systems with short and long routes. In columns 6-8 we present the average required safety stocks per operation, the average buffer sizes required (maximum buffer level throughout the cycle) and the average WIP per operation for the cyclic schedules constructed by simulation using FBDR. All these values are averages over all operations in all the problem instances of the same type.

Problem name	Problem size	$T_{r,o}$ Range	Product types	Products per cycle	Average Safety Stocks	Average Buffer sizes	Average WIP
1	2	3	4	5	6	7	8
abz5	10-10-10	50-99	4.5	8.6	0.85	1.13	0.68
abz6	10-10-10	20-98	7.0	20.1	0.83	1.23	0.72
yn1	20-20-20	10-49	14.1	31.9	0.77	1.20	0.72
yn2	20-20-20	10-49	10.7	24.3	0.80	1.16	0.72
yn3	20-20-20	10-49	11.4	26.5	0.81	1.21	0.74
yn4	20-20-20	10-49	13.3	26.8	0.74	1.14	0.69
pr1	10-20-40	11-49	9.8	24.8	0.75	1.30	0.73
pr2	10-20-40	2-99	10.1	23.9	0.71	1.29	0.70
pr3	10-20-40	2-997	8.3	26.4	0.80	1.44	0.78
ext	10-10-10	1-999	5.5	979.1	10.72	21.16	10.52

Table 3: Summary of the numerical study. Each line presents the averages over 20 problem instances of the same type.

It is well demonstrated by the numerical results that our method is capable of obtaining near optimal solutions for many instances of the problem. For product mixes that allow revenue within one percent of the maximum, we need little safety stocks and buffer spaces and we use very little WIP.

6 Discussion

In this paper we studied the problem of simultaneously selecting a product mix and developed an asymptotically dispatching rule for a jobshop system so as to maximize the long run revenue rate. The obtained optimal solution for the infinite case may serves as a good practical substitution for long planning horizon, especially when its length is unknown in advance.

For problems with large number of jobs we constructed a fluid counterpart system by relaxing the discrete nature of the jobs and allowing each machine to divide its effort among different operations. In large problems, the suggested technique which mimics the operation of the fluid system, enables us to ignore the combinatorial structure of the problem which is the essential difficulty of the problem. The obtained fluid system is simpler, easier to analyze and its optimal revenue serves as an upper bound for the maximal revenue of the discrete jobshop system. This bound is being achieved at steady state. In order to reach this steady state, safety stocks must be built during an initialization phase. We show that our policy creates cyclic schedules and we present some upper bounds on the levels of the safety stocks that are independent of the cycle length. The sizes of the safety stocks are of particular interest due to their effect on the length of the initialization phase as well as on the average WIP after the initialization. These WIP are needed for the system to operate smoothly, but at the same time affects the profit of the system.

The obtained cyclic schedules may consist of arbitrarily large number of jobs that are difficult to handle and can cause high levels of safety stocks and WIP. To overcome this obstacle, we formulate a compact Mixed Integer Program that allows us to reduce the cycle length by slightly compromising on the optimality of the product mix. This gives asymptotically δ-approximation method for the revenue problem with shorter cycles.

As demonstrated by our numerical study for the linear revenue function case, by giving up 1% of the revenue, it is possible to create a Minimal Part Set that consists of very few products. The shorter cycles obtained by this method can be scheduled according to our proposed Fluid Based Dispatching Rule or by any other method in order to reduce the required safety stocks and WIP.

Our results applicable mainly for situations where the product mix should remain fixed over long period. However, an interesting implication of Proposition 3.9 is useful particulary for a make-to-order production mode. Consider a firm that process batches of orders one at a time. Once the processing of a batch is finished a new batch that includes all the orders that were collected since the beginning of the previous batch is started. Let us denote the total number of products of type r in a given batch by n_r and let $C_{max} = \max_{i \in M} \sum_{(r,o) \in \sigma_i} n_r T_{r,o}$ be the congestion on a bottleneck machine. Clearly C_{max} is a lower bound on the time required to process the batch. Now, if the system starts with safety stocks of $B_{r,o}(0) = \lceil \gamma_{r,o-1} \rceil$ it is possible to process the batch according to FBDR, with $x_r = n_r / C_{max}$. The required number of products n_r will be output from the system after C_{max} units of time and the stocks at the system's buffers will return to their initial level by this time. That is, $B_{r,o}(C_{max}) = B_{r,o}(0) = \lceil \gamma_{r,o-1} \rceil$. Hence, at time C_{max} a new batch, of arbitrary composition, can be launched, and so on and so forth. Operating the system in such a mode is likely to be efficient, because the bottleneck machines with respect to the current batch are kept busy uninterruptedly. This operation mode is made possible by Proposition 3.9 provide an upper bound for the required safety stocks independent of the composition of the batch (product mix). However, this efficiency is obtained at the cost of building and maintaining possibly large safety stocks.

We note that our method is applicable to reentrant jobshop that are common in the semiconductors and PCB industries. It can be easily extended to flex-shop, a generalization of jobshop in which each product can be produced using some different routes. For this case, we simply represent each route as a distinct product, solve Linear Program 2.1 and proceed as usual. It may be also possible to extend our method to different production regimes such as, assembly lines, multistage shops and others.

Acknowledgment: Partial support was received from the fund for the promotion of research at the Technion.

References

J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, pages 391–401, 1988.

D. Bertsimas and D. Gammarnik. Asymptotically optimal algorithm for job shop scheduling and packet routing. *Journal of Algorithms*, 33:296–318, 1999.

D. Bertsimas, D. Gammarnik, and J. Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: The holding cost objective. *Operations Research*, 51:798–813, 2003.

D. Bertsimas and J. Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: The makespan objective. *Mathematical Programming*, 92:61–102, 2002.

T. Boudoukh, M. Penn, and G. Weiss. Scheduling job shop with some identical or similar jobs. *Journal of Scheduling*, 4:177–199, 2001.

J.G. Dai and G. Weiss. A fluid heuristic for minimizing makespan in job-shops. *Operations Research*, 50:692–707, 2002.

L. Fleischer and J. Sethuraman. Approximately optimal control of fluid networks. *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, Baltimore, Maryland*, pages 56–65, 2003.

N.G. Hall, T.E. Lee, and M.E. Posner. The complexity of cyclic shop scheduling problems. *Journal of Scheduling*, 5:307–327, 2002.

C. Hanen. Study of an np-hard cyclic scheduling problem: the periodic recurrent job shop. *European Journal of Operational Research*, 72:82–101, 1994.

H. Kamoun and C. Sriskandarajah. The complexity of scheduling jobs in repetitive manufacturing systems. *European Journal of Operational Research.*, 70:350–364, 1993.

T.E. Lee and M.E. Posner. Performance measures and schedule patterns in periodic job shops. *Operations Research*, 45:72–91, 1997.

G. Weiss. A simplex based algorithm to solve separated continuous linear programs. *working paper*, 2001.

T. Yamada and R. Nakano. A genetic algorithm applicable to large-scale job-shop instances. In R. Männer and B. Manderick, editors, *Parallel instance solving from nature 2*, pages 281–290. North-Holand, Elsiver, 1992.

A Appendix - Upper Bound on the Lateness

In this Appendix we provide a proof for proposition 3.7. Our proof is based on demonstrating a stronger bound. This bound is not presented in the main part of the article because of simplicity reasons, mainly those of notations.

We start with a few definitions. Consider a given operation (r, o) carried out by machine $\sigma(r, o)$. Let $\sigma^{fast(r,o)}$ denote the set of all operations carried out by this machine with a production rate of at least x_r, and by $\sigma^{slow(r,o)}$ the complement set of operations. That is, $\sigma^{fast(r,o)} \equiv \{(l,u) \in \sigma(r,o) : x_l \geq x_r\}$ and $\sigma^{slow(r,o)} \equiv \sigma(r,o) \setminus \sigma^{fast(r,o)}$. Recall that $\sigma^{fast(r,o)^-} \equiv \sigma^{fast(r,o)} \setminus \{(r,o)\}$.

Proposition A.1 *Assume a system operates under FBDR while allowing buffer levels to go negative. Then, at any given time t, the lateness is bounded above by*

$$\mathcal{L}_{r,o}(t) \leq \frac{\sum_{(l,u)\in\sigma^{fast(r,o)^-}} \left[T_{l,u} - (x_l - x_r)T_{l,u}^2\right] + \sum_{(l,u)\in\sigma^{slow(r,o)}} (T_{l,u} - x_l T_{l,u}^2)}{\sum_{(l,u)\in\sigma^{fast(r,o)}} T_{l,u}} + x_r T_{r,o}.$$

The proof of the above proposition follows from the following lemma.

Lemma A.2 *Consider a jobshop scheduled according to FBDR. Assume a pair of operation types (l,u) and (r,o), both processed on the same machine, with $x_l \geq x_r$. Then, at any time t, $\mathcal{L}_{l,u}(t) \geq \mathcal{L}_{r,o}(t) - 1 + (x_l - x_r)T_{l,u}$.*

Proof. Let us define the function

$$\Delta^{\mathcal{L}}(t) := \mathcal{L}_{l,u}(t) - \mathcal{L}_{r,o}(t).$$

Then,

$$\Delta^{\mathcal{L}}(t) = t(x_l - x_r) - D_{l,u}(t) + D_{r,o}(t).$$

Since $x_l - x_r \geq 0$, $\Delta^{\mathcal{L}}(t)$ increases everywhere, except at those points of time when operation (l,u) is completed. Hence, local minimum points can be found only at these times. Assume t_0 is such a time and so at time $t_0 - T_{l,u}$ the processing of the operation started. Since the operations are scheduled according to FBDR, $\Delta^{\mathcal{L}}(t_0 - T_{l,u}) \geq 0$. This is because otherwise, operation (l,u) would have not been dispatched at that time since there is at least one operation, operation (r,o), of a greater lateness. Thus,

$$\Delta^{\mathcal{L}}(t_0) = \Delta^{\mathcal{L}}(t_0 - T_{l,u}) - 1 + (x_l - x_r)T_{l,u} \geq -1 + (x_l - x_r)T_{l,u},$$

and we are done. ■

Proof for Proposition A.1: We first consider the lateness at those points of time when machine $\sigma(r,o)$ is ready. By Lemma 3.3, for any given time t,

$$\sum_{(r,o)\in\sigma_i} [t \cdot x_r - D_{r,o}(t)]T_{r,o} = \sum_{(r,o)\in\sigma_i} \mathcal{L}_{r,o}(t)T_{r,o} \leq 0.$$

Thus, for operation (r,o),

$$\sum_{(l,u)\in\sigma(r,o)^-} \mathcal{L}_{l,u}(t)T_{l,u} \leq -\mathcal{L}_{r,o}(t)T_{r,o}, \tag{9}$$

and (9) can be rewritten as

$$\sum_{(l,u)\in\sigma^{fast(r,o)^-}} \mathcal{L}_{l,u}(t)T_{l,u} + \sum_{(l,u)\in\sigma^{slow(r,o)}} \mathcal{L}_{l,u}(t)T_{l,u} \le -\mathcal{L}_{r,o}(t)T_{r,o}.$$

By Lemma 3.11[1] , $\mathcal{L}_{l,u}(t) \ge x_l T_{l,u} - 1$ and by Lemma A.2 all the operations $(l,u) \in \sigma^{fast(r,o)}$ admit $\mathcal{L}_{l,u}(t) \ge \mathcal{L}_{r,o}(t) - 1 + (x_l - x_r)T_{l,u}$. Hence,

$$\sum_{(l,u)\in\sigma^{fast(r,o)^-}} [\mathcal{L}_{r,o}(t) - 1 + (x_l - x_r)T_{l,u}]T_{l,u} + \sum_{(l,u)\in\sigma^{slow(r,o)}} [x_l T_{l,u} - 1]T_{l,u} \le -\mathcal{L}_{r,o}(t)T_{r,o},$$

implying that,

$$\sum_{(l,u)\in\sigma^{fast(r,o)}} \mathcal{L}_{r,o}(t)\cdot T_{l,u} \le \sum_{(l,u)\in\sigma^{fast(r,o)^-}} \left[T_{l,u} - (x_l - x_r)T_{l,u}^2\right] + \sum_{(l,u)\in\sigma^{slow(r,o)}} (T_{l,u} - x_l T_{l,u}^2).$$

Thus, for any time t when machine $\sigma(r,o)$ is ready, we have ,

$$\mathcal{L}_{r,o}(t) \le \frac{\sum_{(l,u)\in\sigma^{fast(r,o)^-}} \left[T_{l,u} - (x_l - x_r)T_{l,u}^2\right] + \sum_{(l,u)\in\sigma^{slow(r,o)}} (T_{l,u} - x_l T_{l,u}^2)}{\sum_{(l,u)\in\sigma^{fast(r,o)}} T_{l,u}}. \tag{10}$$

This inequality is true for any operation when a machine is ready, i.e., idle or about to start a new operation. Clearly if the machine is idle then the lateness of all the operations is negative and our bound is satisfied. If the machine starts a new operation at time t, let us denote this operation by (r', o'). Recall that by the definition of FBDR, at time t, operation (r', o'), maximizes the lateness over all the operations of machine $M_{r',o'}$. During the processing of (r', o') it accumulates additional lateness of $x_r' T_{r',o'}$ and hence satisfies our bound. Now, by the first part of our proof, the rest of the operations on machine $M_{r',o'}$ must satisfy (10) at the next time that the machine is ready (time $t + T_{r',o'}$) and so, since their lateness increase throughout the interval $[t, t + T_{r',o'}]$ it must satisfy (10) during this interval and our bound follows. ■

Proof of Proposition 3.7. We start with the tighter bound obtained in Proposition A.1.

$$\mathcal{L}_{r,o}(t) \le \frac{\sum_{(l,u)\in\sigma^{fast(r,o)^-}} \left[T_{l,u} - (x_l - x_r)T_{l,u}^2\right] + \sum_{(l,u)\in\sigma^{slow(r,o)}} (T_{l,u} - x_l T_{l,u}^2)}{\sum_{(l,u)\in\sigma^{fast(r,o)}} T_{l,u}} \le$$

$$\frac{\sum_{(l,u)\in\sigma(r,o)^-} T_{l,u}}{\sum_{(l,u)\in\sigma^{fast(r,o)}} T_{l,u}} \le \frac{\sum_{(l,u)\in\sigma(r,o)^-} T_{l,u}}{T_{r,o}}.$$

[1]Although this proposition is used to prove Proposition 3.7 there is no circularity here since our proof of Lemma 3.11 does not rely on anything derived from Proposition 3.7.

The inequalities above follow directly from the definitions of the sets $\sigma(r,o)^-$,$\sigma^{fast(r,o)}$, $\sigma^{slow(r,o)}$, $\sigma^{fast(r,o)^-}$, and $\sigma^{slow(r,o)^-}$. ■

We note that the weaker bound of proposition 3.7 demonstrates the fact that the lateness (and thus the required safety stocks and WIP) can be bounded by a finite number that is independent of the particular optimal product mix obtained.

B Appendix - List of Notations

$\lceil a \rceil$	the smallest integer that is not smaller than a
$\lfloor a \rfloor$	the greatest integer that is not greater than a
$B_{r,o}(t)$	number of jobs that already completed operation $(r, o-1)$ at time t but are yet to start operation (r,o)
$B^+_{r,o}(t)$	number of jobs that already completed operation $(r, o-1)$ at time t but are yet to complete operation (r,o)
$D_{r,o}(t)$	number of jobs that by time t already completed operation $(r, o-1)$ but are yet to start operation (r,o)
j_r	number of products of type r per cycle in a cyclic schedule
K_r	number of operations in route r
$\mathcal{L}_{r,o}(t)$	$t \cdot x_r - D_{r,o}(t)$, the lateness of operation (r,o) at time t
M	set of machines
$M_{r,o}$	the machine that processes operation (r,o)
(r,o)	o^{th} operations of product type r
R	set of product types (routes)
$T_{r,o}$	processing time of operation (r,o)
x_r (x^*_r)	the processing rate of all the operations in route r according to some (optimal) solution of the fluid problem
$\gamma_{r,o}$	an upper bound on the lateness of operation (r,o)
σ_i	set of operations carried out by machine i
$\sigma(r,o)$	set of all operations carried out on the same machine as operation (r,o)
$\sigma(r,o)^-$	all operations $\sigma(r,o)$ excluding (r,o) itself
$\sigma^{fast(r,o)}$	all operations $\sigma(r,o)$ with their production rate at least that of the production rate of (r,o)
$\sigma^{fast(r,o)^-}$	same as $\sigma^{fast(r,o)}$ excluding operation (r,o)
$\sigma^{slow(r,o)}$	all the operations $\sigma(r,o)$ with production rate smaller than the production rate of (r,o)